哈尔滨工业大学（深圳）

大数据导论大作业报告

题目：特定疾病的回归和分类

姓　　名　　郑漫莎

学　　号　　180110711

报告日期　　2021/1/8

一、 实验目的

对实验进行概括描述，并分析一下涉及到的技术要点，说明整个实验将以怎样的顺序进行等。

整个实验分为两部分，对患病指标值的预测和是否患病的预测。

1. 对患病指标值的预测：
   a) 读入数据，对数据进行预处理。
   b) 将于处理后的数据进行划分。
   c) 构建模型，进行模型训练和测试
   d) 根据均方误差对结果进行评估
   e) 调参
2. 对是否患病的预测：
   a) 读入数据，对数据进行预处理。
   b) 将处理后的数据进行划分。
   c) 构建模型并训练和测试
   d) 根据 F1 对结果进行评估
   e) 调参

二、 实验内容分析

对实验进行概括描述，并分析一下涉及到的技术要点，说明整个实验将以怎样的顺序进行等。

1. 对患病指标的预测：回归
   a) 文件读取：根据终端输入进行读取文件的判断
   b) 数据预处理：
      i. 数据填充：乙肝相关数据正常人为 0，故缺失值用 0 填充。其他数据采用平均值填充
      ii. 将字符型数据转为数值型数据：年龄
      iii. 数据离散化：年龄根据数据分布，在两端数据密度低的位置距离分箱间隔较大，中间数据密度高的位置分箱间隔较小。
      iv. 日期的数值转化：观察数据发现数据都为 2017 年数据，故将日期对应为距离 2018.1.1 的天数
      v. 数据归一化：为减少数据大小对结果的影响，将部分数据归一化为 0-1 之间
   c) 数据集划分：大部分为训练集，小部分测试集
   d) 构建模型，进行模型训练和测试：训练集训练，测试集测试
      i. 模型采用回归相关模型，根据模型分别测试，选择线性回归模型效果最佳
   e) 计算测试集正确结果和测试结果的均方误差，观察均方误差
   f) 根据均方误差进行参数调整
2. 对是否患病的预测：分类
   a) 文件读取：根据终端输入进行读取文件的判断
   b) 对数据进行预处理：
      i. 数据类型冗余和删除：id 对结果无关；BMI 分类是有身高和体重计算而来，同时进行了数据离散化和分类，故身高和体重数据无关

ii. 数据缺失值填充：孕次、产次、家族史、ACEID 用 0 填充，BMI 用众数填充，其余用平均值填充

iii. 数据离散化：年龄离散化

iv. 数据归一化：为减少数据大小对结果的影响，将部分数据归一化为 0-1 之间

c) 将处理后的数据进行划分：训练集较多，测试集较少

d) 构建模型，进行模型训练和测试：训练集训练，测试集测试

i. 模型采用决策树模型

e) 根据 F1 对结果进行评估

f) 调参

三、 实验过程及结果

包括算法实现的主要步骤，算法实现的关键代码，算法运行结果截图，算法性能曲线图及结果分析等。

D_model.py：

1. 主要步骤：数据读取、数据预处理、数据划分、模型训练、结果分析
2. 数据读取：
   a) Sys.argv 获取程序运行前输入的文件，如果无输入，则在程序运行过程中从终端

```
if len(sys.argv) == 2:
    s = sys.argv[1]
else:
    s = input("文件位置：")
```

3. 数据预处理：
   a) 数据初始化：
      i. 根据上述文件位置 s，读取数据

```
f = open(file)
self.df = pandas.read_csv(f)
cols = self.df.columns.values
```

      ii. 将数据转化为二维数组格式

```
df_array = []
for i in range(len(self.df)):
    df_array.append(self.df.iloc[i].values)
df_array = numpy.array(df_array)
```

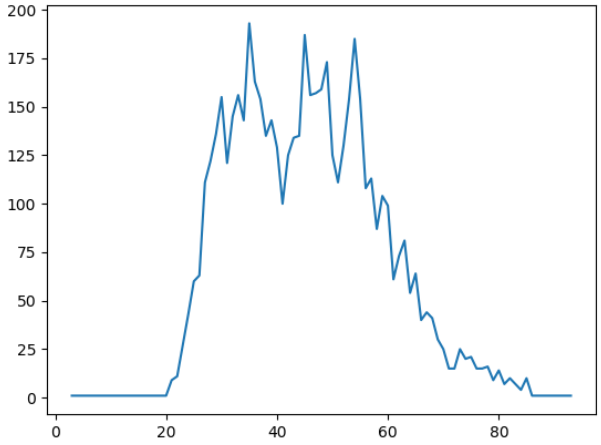
      iii. 数据转化为 DataFrame 格式

```
df_frame = {}
for i in range(len(cols)):
    df_frame[cols[i]] = df_array[:, i]
self.df = pandas.DataFrame(df_frame)
```

   b) 数据预处理：
      i. 查看数据可知，性别、年龄、日期无缺失

      ii. 性别转换为数值型：男 0，女 1

      iii. 年龄根据数据知最小 3，最大 93，根据折线图，在两端划分间隔长，中间间隔短，进行分箱操作




```
self.df["年龄"] = self.df["年龄"].astype(int)
 bins = [2, 20, 30, 40, 50, 60, 70, 94]
self.df["年龄"] = pandas.cut(self.df["年龄"], bins, labels=False)
```

iv. 日期转为距离 2018.1.1 的天数，由于体检日期都为 2017 年

```
self.df["体检日期"] = pandas.to_datetime(self.df['体检日期'], format="%d/%m/%Y")
self.df["体检日期"] = pandas.to_datetime('1/1/2018', format="%d/%m/%Y") - self.df["体检日期
"]
self.df["体检日期"] = self.df["体检日期"].dt.days
```

v. 缺失值填充：

1. 乙肝相关信息：0 填充，正常人没有乙肝相关抗原抗体
2. 其余信息：平均值填充

vi. 归一化处理：

```
for line in self.df.iloc[range(len(self.df)), range(4, 41)]:
    if "乙肝" in line:
        self.df[line] = self.df[line].fillna(0)
    else:
        self.df[line] = self.df[line].fillna(self.df[line].mean())
    self.df[line] = (self.df[line] - self.df[line].min()) / (self.df[line].max() -
self.df[line].min())
```

4. 数据划分：
   a) 数据整体大小为 5642，其前 5000 位训练集，其余为测试集
   b) 重新读取数据时，忽略 id，id 对结果无影响

```
X_train = X[:5000]
y_train = y[:5000]
X_test = X[5000:]
y_test = y[5000:]
```

5. 模型选择和训练：
   a) 由于该问题为回归问题，预测选择模型为线性回归模型，决策树模型、逻辑回归模型。
   b) 调用相关库函数进行训练和测试
6. 结果分析：
   a) 用均方误差估计结果数值

```
sum_result = 0
```

```
for i in range(len(y_test)):
    sum_result += pow(pre_y[i]-y_test[i], 2)
sum_result /= (2 * len(y_test))
print(sum_result)
```

b) 打印拟合图形

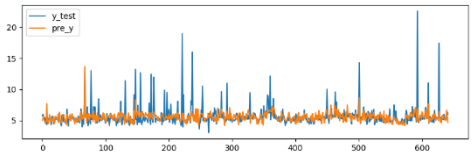
7. 结果：

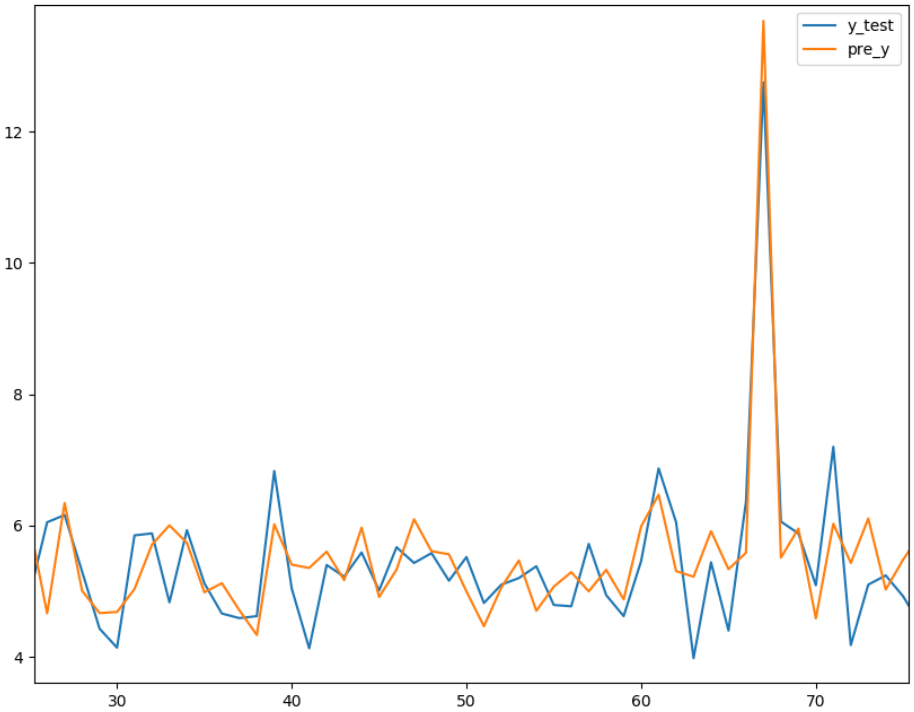
a) 根据个模型的军方结果，最后模型确定为线性回归模型。结果在可接受范围之内，对于血糖过大的值拟合效果较差

b) 均方误差：

```
1.0936581453896752
```

c) 拟合图形





F_model.py：

1. 主要步骤：数据读取、数据预处理、数据划分、模型训练、结果分析
2. 数据读取：

a) Sys.argv 获取程序运行前输入的文件，如果无输入，则在程序运行过程中从终端

```
if len(sys.argv) == 2:
    s = sys.argv[1]
else:
    s = input("文件位置：")
```

3. 数据预处理：
   a) 数据初始化：
      i. 根据上述文件位置 s，读取数据

```
f = open(file)
self.df = pandas.read_csv(f)
cols = self.df.columns.values
```

      ii. 将数据转化为二维数组格式

```
df_array = []
for i in range(len(self.df)):
    df_array.append(self.df.iloc[i].values)
df_array = numpy.array(df_array)
```

      iii. 数据转化为 DataFrame 格式

```
df_frame = {}
for i in range(len(cols)):
    df_frame[cols[i]] = df_array[:, i]
self.df = pandas.DataFrame(df_frame)
```

   b) 数据预处理：
      i. 无关数据删除：
         1. Id 和结果无关
         2. BMI 分类代表了身高和体重的计算和分类，身高和体重因素冗余

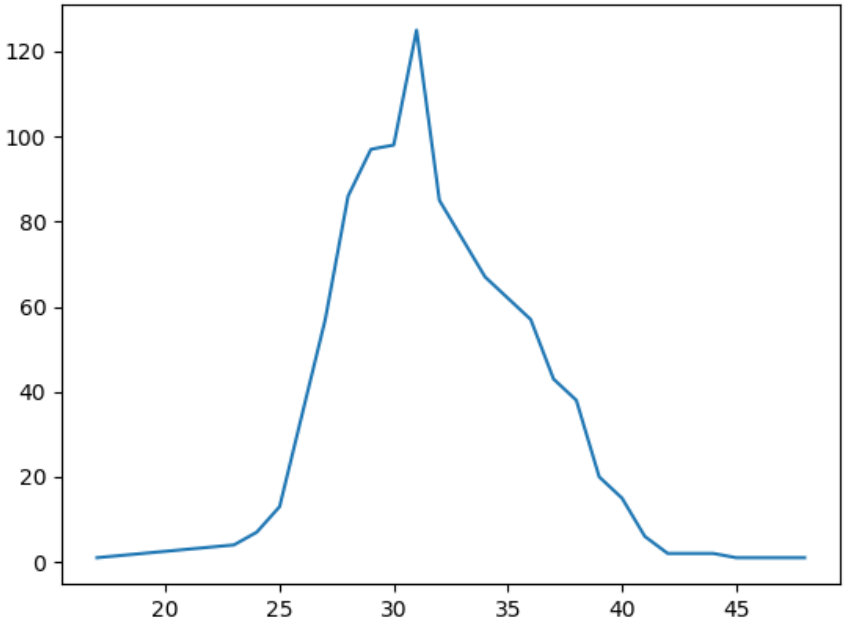
```
del(self.df["id"])
del(self.df["身高"])
del(self.df["孕前体重"])
```

      ii. 缺失值填充：
         1. 0 填充：
            a) SNP 相关属性缺失值单独为一类记为 0
            b) 孕次和产次正常记为 0
            c) DM 家族史和 ACEID，RBP4 正常无，记为 0
         2. 众数填充：
            a) BMI 分类：正常范围在正常人中间类型
         3. 平均数填充：
            a) 其余属性用平均数填充

```
for line in self.df.columns:
    # SNP 缺失值等 0 填充
    if "SNP" in line or line in ["孕次", "产次", "DM 家族史", "ACEID"]:
        self.df[line] = self.df[line].fillna(0).astype(int)
    # 年龄身高等用平均数填充
    if line in ["年龄", "收缩压", "舒张压", "ALT", "AST"]:
        self.df[line] = self.df[line].fillna(self.df[line].mean()).astype(int)
    if line in ["孕前 BMI", "分娩时", "糖筛孕周", "VAR00007", "wbc"]\
            or line in self.df.columns[37:47]:
        self.df[line] = self.df[line].fillna(self.df[line].mean())
    # BMI 分类用众数填充
    if line == "BMI 分类":
```

```
        self.df[line] = self.df[line].fillna(self.df[line].mode()[0]).astype(int)
# RBP40 填充
self.df["RBP4"] = self.df["RBP4"].fillna(0)
```

iii. 年龄根据数据知最小 17，最大 48，根据折线图，在两端划分间隔长，中间间隔短，进行分箱操作




```
self.df["年龄"] = self.df["年龄"].astype(int)
bins = [16, 25, 30, 35, 40, 50]
self.df["年龄"] = pandas.cut(self.df["年龄"], bins, labels=False)
```

iv. 归一化处理：

```
for line in self.df.columns:
    if line in ["RBP4", "孕前 BMI", "收缩压", "舒张压", "分娩时",
                "糖筛孕周", "VAR00007", "wbc", "ALT", "AST"]\
            or line in self.df.columns[37:47]:
        self.df[line] = (self.df[line] - self.df[line].min()) / (self.df[line].max() -
self.df[line].min())
```

4. 数据划分：
   a) 数据整体大小为 1000，其前 900 位训练集，其余为测试集
   b) 重新读取数据时，忽略 id，id 对结果无影响

```
X_train = X[:900]
y_train = y[:900]
X_test = X[900:]
y_test = y[900:]
```

5. 模型选择和训练：
   a) 由于该问题为分类问题，预测选择模型选择为决策树模型
   b) 调用相关库函数进行训练和测试
6. 结果分析：
   a) 用 F1 估计结果数值

```
correct1 = 0
```

```
for i in range(len(y_test)):
    if y_test[i] == 1 and y_test[i] == pre_y[i]:
        correct1 += 1
correct_rate = correct1 / pre_y.sum()
recall_rate = correct1 / y_test.sum()
print((2 * correct_rate * recall_rate) / (correct_rate + recall_rate))
```

b) 打印拟合图形

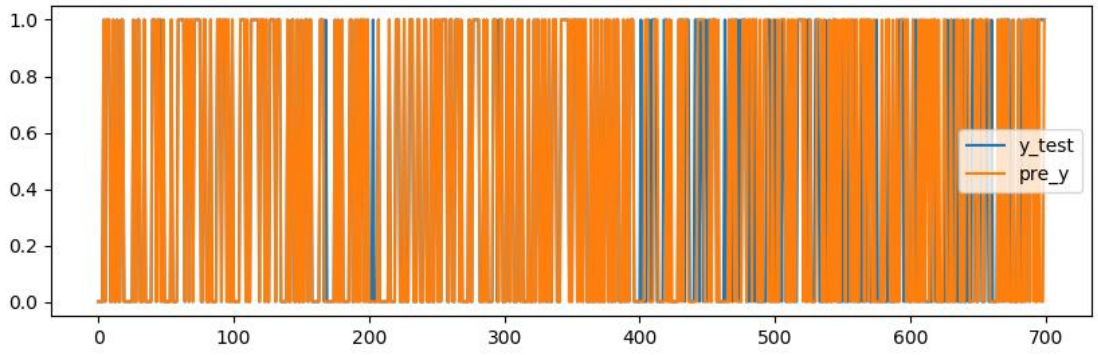
7. 结果：

a) 结果在可接受范围之内，由于数据选择是前后分段，没有随机选择，对于部分区域数据拟合效果较差

b) F1：

```
C:\ProgramData\Anaconda3\envs\nlp\python.exe D:/zms/big_data/f_model.py
0.8146964856230032
```

c) 拟合图形



四、 实验心得

实验完成后的感悟与总结。

大数据最开心最难过的就是调参，最后的调参变成了不知所谓的遍历，选择最好的值。最要脑子的应该是数据的预处理，一个数据用什么填缺失值，正常应该是什么，缺失值的多少都会影响数据的处理结果，数据的归一化和分箱什么的。第二个实验感觉很多数据是相关联的，但是不知道怎么把他分离开，虽然懂一点分离的理论，利用卡方测相关度什么的，但是在上手方面还是有所缺陷，不太能尝试的出来。